Project2Task0:

🏁Project2Task0Client:

```
package edu.cmu.ds;

import java.net.*;
import java.io.*;
import java.util.Scanner;

public class EchoClientUDP{
    /**
     * Initialize the DatagramSocket
     * This is a Java program for a UDP client that sends messages to a
server and receives echoed messages back.
     * The program takes input from the user, sends it to the server on a
specified port number,
     * receives a reply from the server, and prints the reply to the console.
     * @param args
     */
    public static void main(String args[]){
        // args give message contents and server hostname
        // Here we initialize the DatagramSocket
        System.out.println("The UDP client is running.");
        DatagramSocket aSocket = null;
        try {
            // Initialize the InetAddress with the hostname "localhost"
            InetAddress aHost = InetAddress.getByName("localhost");
            // Assign the server port to the variable "serverPort"
            // Creates a new scanner object that reads from System.in
            Scanner scanner = new Scanner(System.in);
            // Print Tips.
            System.out.println("Enter server port number: ");
            // Set portNum
            int serverPort = scanner.nextInt();
            // Initialize the DatagramSocket object "aSocket"
            aSocket = new DatagramSocket();
            System.out.println("Connect the server in port number: " +
serverPort);
            // Initialize the BufferedReader "typed" to read from standard
input
            BufferedReader typed = new BufferedReader(new
InputStreamReader(System.in));
            // Start a loop that runs until the user stops entering input
            String nextLine;
            while ((nextLine = typed.readLine()) != null) {
                // Convert the input string to a byte array
                byte [] m = nextLine.getBytes();
                // Create a DatagramPacket "request" with the message and the
server address/port
                DatagramPacket request = new DatagramPacket(m, m.length,
aHost, serverPort);
                // Send the request packet using the DatagramSocket "aSocket"
                aSocket.send(request);
                // Initialize a buffer to receive the reply
```

```
                byte[] buffer = new byte[1000];
                // Create a DatagramPacket "reply" to hold the reply
                DatagramPacket reply = new DatagramPacket(buffer,
buffer.length);
                // Receive the reply packet using the DatagramSocket
"aSocket"
                aSocket.receive(reply);
                // Get requestString data
                String requestString = new String(reply.getData(), 0,
reply.getLength());
                // Check if string equals "halt!"
                if (requestString.equals("halt!")) {
                    // Generate datagramPacket response
                    DatagramPacket response = new
DatagramPacket(request.getData(), request.getLength(),
request.getAddress(),request.getPort());
                    // Send response
                    aSocket.send(response);
                    // Print reminder
                    System.out.println("UDP Client side quitting");
                    break;
                }
                // Set the string in correct length
                String replyString = new String(reply.getData(), 0,
reply.getLength());
                // Print the reply message to the console
                System.out.println("Reply from server: " +  replyString);
            }
        // Catch a SocketException if one occurs and print the error message
to the console
        }catch (SocketException e) {System.out.println("Socket Exception: " +
e.getMessage());
        // Catch an IOException if one occurs and print the error message to
the console
        }catch (IOException e){System.out.println("IO Exception: " +
e.getMessage());
        // Finally, close the DatagramSocket if it is not null
        }finally {if(aSocket != null) aSocket.close();}
    }
}
```

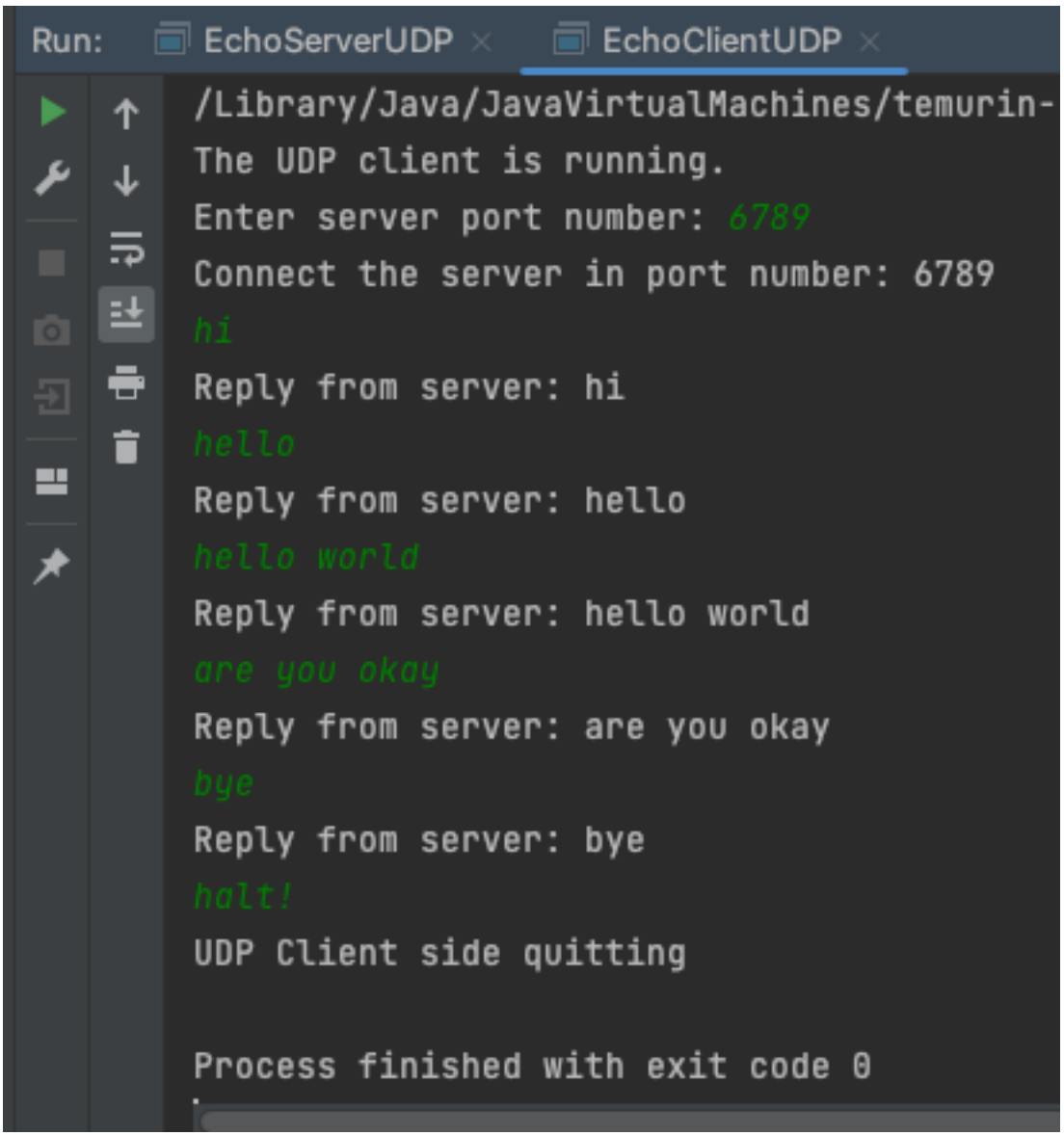
🏁Project2Task0Server

```
package edu.cmu.ds;

import java.net.*;
import java.io.*;
import java.util.Scanner;

public class EchoServerUDP{
    /**
     * this code sets up a simple UDP server that listens for incoming
packets on port 6789,
     * echoes the contents of each packet back to the client,
     * and prints the contents of each packet to the console.
     * The server runs in an infinite loop until it is terminated.
```

```
     * @param args
     */
    public static void main(String args[]){
        System.out.println("The UDP server is running.");
        DatagramSocket aSocket = null;
        byte[] buffer = new byte[1000];
        try{
            Scanner scanner = new Scanner(System.in);
            System.out.print("Enter server port number: ");
            int serverPort = scanner.nextInt();
            System.out.println("The server is listen on the port number: " +
serverPort);
            aSocket = new DatagramSocket(serverPort);
            // create a new DatagramPacket object for incoming requests
            DatagramPacket request = new DatagramPacket(buffer,
buffer.length);
            // Set a variable to control the while loop
            Boolean halt = false;
            while(!halt){
                // wait for incoming requests
                aSocket.receive(request);
                // Initialize a requestString message
                String requestString = new String(request.getData(), 0,
request.getLength());
                // If String is "halt!"
                if (requestString.equals("halt!")) {
                    // Generate datagramPacket response
                    DatagramPacket response = new
DatagramPacket(request.getData(), request.getLength(),
request.getAddress(),request.getPort());
                    // Send response
                    aSocket.send(response);
                    // Try to receive the "halt!" message
                    aSocket.receive(request);
                    String haltString = new String(request.getData(), 0,
request.getLength());
                    System.out.println(haltString);
                    if (haltString.equals("halt!")) {
                        aSocket.send(response);
                        // Stop while loop
                        halt = true;
                    }
                } else {
                    // create a new DatagramPacket object for the reply
                    DatagramPacket reply = new
DatagramPacket(requestString.getBytes(),
                            requestString.getBytes().length,
request.getAddress(), request.getPort());
                    // print the request string to the console
                    System.out.println("Echoing: " + requestString);
                    // send the reply to the client
                    aSocket.send(reply);
                }
            }
            System.out.println("UDP Server side quitting");
            // Catch a SocketException if one occurs and print the error
message to the console
```

```
        }catch (SocketException e) {System.out.println("Socket Exception: " +
e.getMessage());
            // Catch an IOException if one occurs and print the error message
to the console
        }catch (IOException e){System.out.println("IO Exception: " +
e.getMessage());
            // Finally, close the DatagramSocket if it is not null
        }finally {if(aSocket != null) aSocket.close();}
    }
}
```

🏁Project2Task0ClientConsole



🏁Project2Task0ServerConsole

Run: EchoServerUDP EchoClientUDP

```
/Library/Java/JavaVirtualMachines/temurin-1
The UDP server is running.
Enter server port number: 6789
The server is listen on the port number: 67
Echoing: hi
Echoing: hello
Echoing: hello world
Echoing: are you okay
Echoing: bye
halt!
UDP Server side quitting

Process finished with exit code 0
```

Project2Task1:

🏁EavesdropperUDP.java

```
package edu.cmu.ds;

import java.net.*;
import java.io.*;
import java.util.Scanner;

public class EchoClientUDP{
    /**
     * Initialize the DatagramSocket
     * This is a Java program for a UDP client that sends messages to a
server and receives echoed messages back.
     * The program takes input from the user, sends it to the server on a
specified port number,
     * receives a reply from the server, and prints the reply to the console.
     * @param args
     */
    public static void main(String args[]){
        // args give message contents and server hostname
        // Here we initialize the DatagramSocket
        System.out.println("The UDP client is running.");
        DatagramSocket aSocket = null;
        try {
            // Initialize the InetAddress with the hostname "localhost"
            InetAddress aHost = InetAddress.getByName("localhost");
            // Assign the server port to the variable "serverPort"
            // Creates a new scanner object that reads from System.in
            Scanner scanner = new Scanner(System.in);
            // Print Tips.
            System.out.print("Enter server port number: ");
            // Set portNum
            int serverPort = scanner.nextInt();
            // Initialize the DatagramSocket object "aSocket"
            aSocket = new DatagramSocket();
            System.out.println("Connect the server in port number: " +
serverPort);
            // Initialize the BufferedReader "typed" to read from standard
input
            BufferedReader typed = new BufferedReader(new
InputStreamReader(System.in));
            // Start a loop that runs until the user stops entering input
            String nextLine;
            while ((nextLine = typed.readLine()) != null) {
                // Convert the input string to a byte array
                byte [] m = nextLine.getBytes();
                // Create a DatagramPacket "request" with the message and the
server address/port
                DatagramPacket request = new DatagramPacket(m, m.length,
aHost, serverPort);
                // Send the request packet using the DatagramSocket "aSocket"
                aSocket.send(request);
```

```
                // Initialize a buffer to receive the reply
                byte[] buffer = new byte[1000];
                // Create a DatagramPacket "reply" to hold the reply
                DatagramPacket reply = new DatagramPacket(buffer,
buffer.length);
                // Receive the reply packet using the DatagramSocket
"aSocket"
                aSocket.receive(reply);
                // Get requestString data
                String requestString = new String(reply.getData(), 0,
reply.getLength());
                // Check if string equals "halt!"
                if (requestString.equals("halt!")) {
                    // Generate datagramPacket response
                    DatagramPacket response = new
DatagramPacket(request.getData(), request.getLength(),
request.getAddress(),request.getPort());
                    // Send response
                    aSocket.send(response);
                    // Print reminder
                    System.out.println("UDP Client side quitting");
                    break;
                }
                // Set the string in correct length
                String replyString = new String(reply.getData(), 0,
reply.getLength());
                // Print the reply message to the console
                System.out.println("Reply from server: " +  replyString);
            }
        // Catch a SocketException if one occurs and print the error message
to the console
        }catch (SocketException e) {System.out.println("Socket Exception: " +
e.getMessage());
        // Catch an IOException if one occurs and print the error message to
the console
        }catch (IOException e){System.out.println("IO Exception: " +
e.getMessage());
        // Finally, close the DatagramSocket if it is not null
        }finally {if(aSocket != null) aSocket.close();}
    }
}
```

🏁Project2Task1ThreeConsoles

```
Run:  EavesdropperUDP  EchoServerUDP  EchoClientUDP
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Hom
The UDP server is running.
Enter server port number: 6789
The server is listen on the port number: 6789
Echoing: hi!
Echoing: hello!
Echoing: are you okay!
Echoing: fine!
halt!
UDP Server side quitting

Process finished with exit code 0
```

```
Run:  EavesdropperUDP  EchoServerUDP  EchoClientUDP
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/b
The UDP client is running.
Enter server port number: 6798
Connect the server in port number: 6798
hi
Reply from server: hi
hello
Reply from server: hello
are you okay
Reply from server: are you okay
fine
Reply from server: fine
halt!
UDP Client side quitting

Process finished with exit code 0
```

Run: EavesdropperUDP EchoServerUDP EchoClientUDP

```
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java
The server is listening on port number: 6798
Received message from client: hi
Sent message to server: hi!
Received response from server: hi!
Sent message to client: hi
Received message from client: hello
Sent message to server: hello!
Received response from server: hello!
Sent message to client: hello
Received message from client: are you okay
Sent message to server: are you okay!
Received response from server: are you okay!
Sent message to client: are you okay
Received message from client: fine
Sent message to server: fine!
Received response from server: fine!
Sent message to client: fine
```

Run: EavesdropperUDP EchoServerUDP EchoClientUDP

```
Sent message to client: hello
Received message from client: are you okay
Sent message to server: are you okay!
Received response from server: are you okay!
Sent message to client: are you okay
Received message from client: fine
Sent message to server: fine!
Received response from server: fine!
Sent message to client: fine
Received message from client: halt!
Sent message to server: halt!
Received response from server: halt!
Sent message to client: halt!
Received message from client: halt!
Sent message to server: halt!
Received response from server: halt!
Sent message to client: halt!
```

Project2Task2:

🏁 Project2Task2Client

```
package edu.cmu.ds;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;

public class AddingClientUDP {
    /**
     * Client component that sends integer values to the server,
     * receives the sum of those integers computed by the server,
     * and displays the result to the user.
     * The client reads user input from the console,
     * and sends requests to the server by creating a DatagramPacket
containing the user input as a string,
     * and sends it to the server at the specified port. The server receives
the packet, parses the integer value,
     * computes the sum, and sends it back to the client. The client then
receives the response from the server
     * and displays it to the user.
     */
    public static void main(String[] args) {
        // Read input from the user
        BufferedReader inputReader = new BufferedReader(new
InputStreamReader(System.in));

        int serverPort = 0;
        try {
            System.out.println("The client is running.");
            System.out.print("Please enter server port: ");
            // Read the server port number from the user input
            serverPort = Integer.parseInt(inputReader.readLine());
        } catch (NumberFormatException e) {
            System.out.println("Invalid port number: " + e.getMessage());
            System.exit(1);
        } catch (IOException e) {
            System.out.println("IO Exception: " + e.getMessage());
            System.exit(1);
        }
        System.out.println(" ");
        // Loop until user enters "halt!"
        while (true) {
            try {
                // Read input from the user
                String inputString = inputReader.readLine();
```

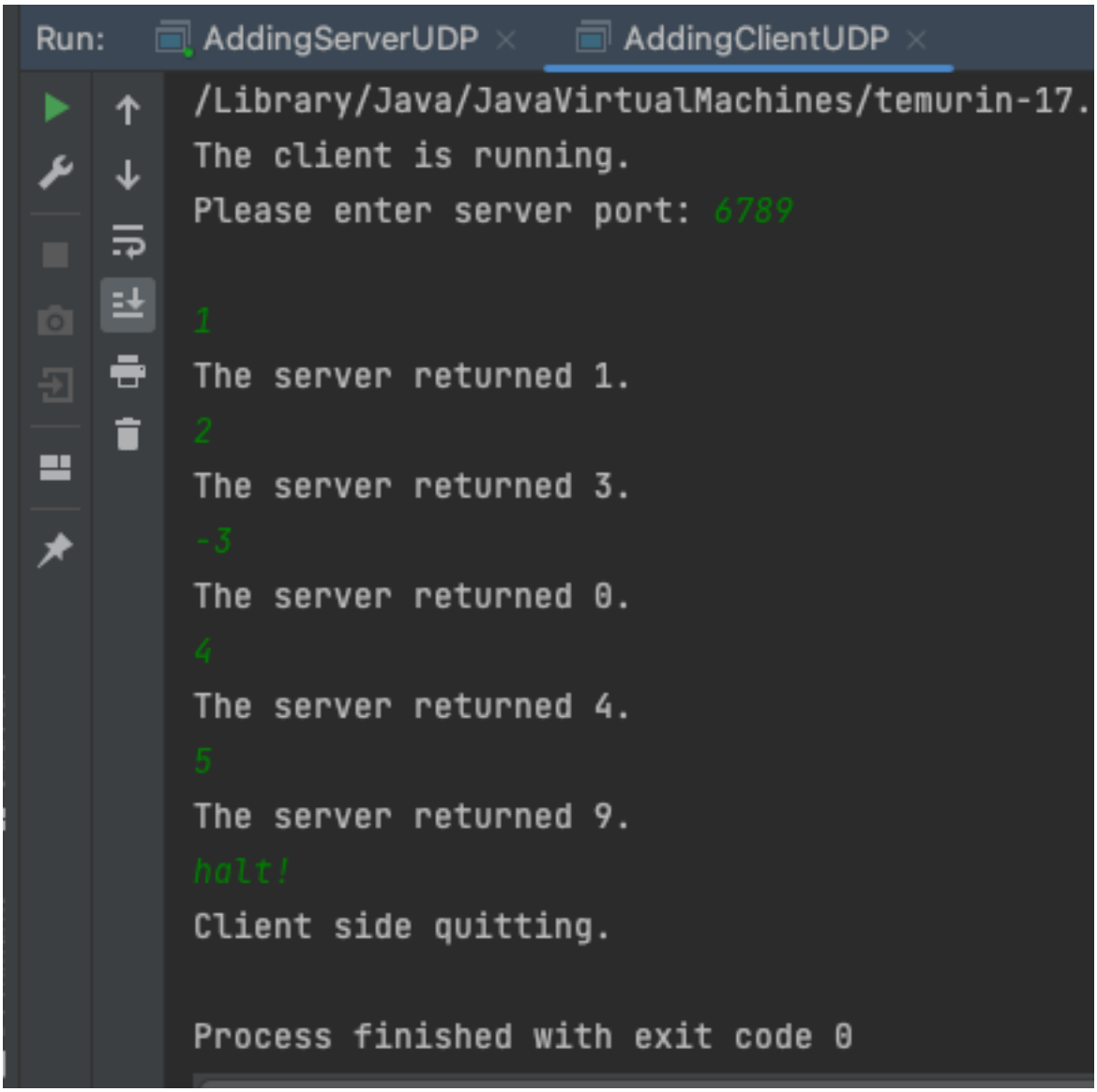
```
                if (inputString.equals("halt!")) {
                    System.out.println("Client side quitting.");
                    break;
                }
                // Convert user input to integer
                int inputInt = Integer.parseInt(inputString);
                // Send request to server and receive response
                int sum = add(inputInt, serverPort);
                System.out.println("The server returned " + sum + ".");
            } catch (NumberFormatException e) {
                System.out.println("Invalid input: " + e.getMessage());
            } catch (IOException e) {
                System.out.println("IO Exception: " + e.getMessage());
            }
        }
    }

    public static int add(int i, int serverPort) throws IOException {
        // Method to encapsulate the socket communication with the server
        final int BUFFER_SIZE = 1000;
        byte[] buffer = new byte[BUFFER_SIZE];
        DatagramSocket clientSocket = new DatagramSocket();
        InetAddress serverAddress = InetAddress.getByName("localhost");
        // Send request to server
        String requestString = Integer.toString(i);
        DatagramPacket requestPacket = new
DatagramPacket(requestString.getBytes(), requestString.getBytes().length,
serverAddress, serverPort);
        clientSocket.send(requestPacket);
        // Receive response from server
        DatagramPacket responsePacket = new DatagramPacket(buffer,
BUFFER_SIZE);
        clientSocket.receive(responsePacket);
        String responseString = new String(responsePacket.getData(), 0,
responsePacket.getLength());
        int sum = Integer.parseInt(responseString);
        // Close the socket and return the sum
        clientSocket.close();
        return sum;
    }
}
```

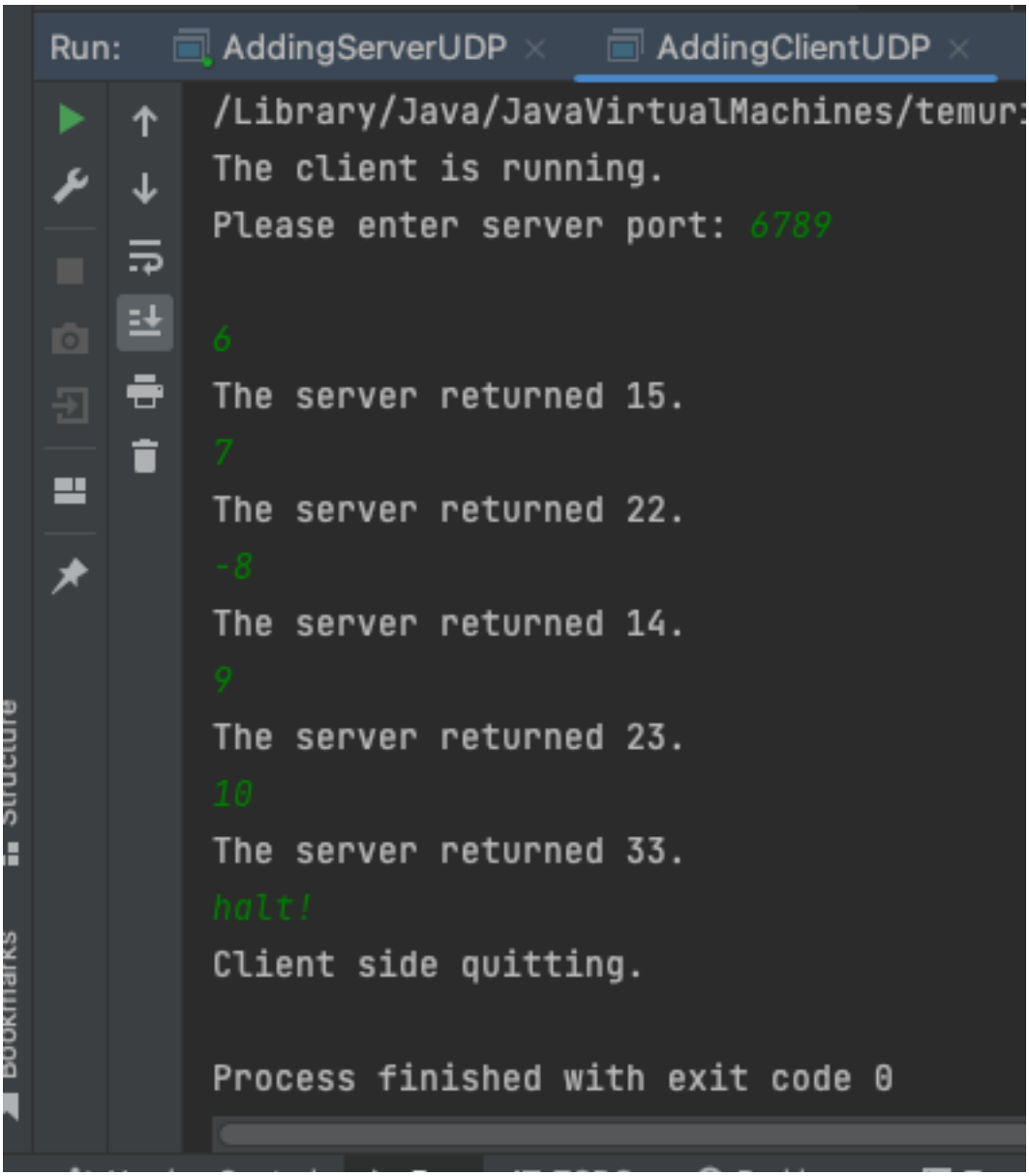
🏁 Project2Task2Server

```
package edu.cmu.ds;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.SocketException;

public class AddingServerUDP {
    /**
     * The main method that creates a server socket, listens for incoming
requests,
```

```java
     * updates the sum with the integer
     * input from the clients, and returns the updated sum to the clients.
     */
    public static void main(String[] args) {
        // The server listens on this port for incoming requests from clients
        final int SERVER_PORT = 6789;
        // The maximum size of the byte array used for packet transmission
        final int BUFFER_SIZE = 1000;
        // The current sum of all integer inputs received from clients
        int sum = 0;
        // The buffer used to store incoming and outgoing packets
        byte[] buffer = new byte[BUFFER_SIZE];
        // The UDP socket used to listen for incoming packets
        DatagramSocket serverSocket = null;

        try {
            // Create a new DatagramSocket and bind it to the specified port
            serverSocket = new DatagramSocket(SERVER_PORT);
            System.out.println("Server started");

            while (true) {
                // Create a new packet to store incoming data
                DatagramPacket requestPacket = new DatagramPacket(buffer,
BUFFER_SIZE);
                // Wait for incoming packets
                serverSocket.receive(requestPacket);
                // Extract the integer value from the incoming packet data
                String requestString = new String(requestPacket.getData(), 0,
requestPacket.getLength());
                int requestInt = Integer.parseInt(requestString);
                System.out.println("Adding: " + requestInt + " to " + sum);
                // Update the sum with the incoming integer value
                sum += requestInt;
                // Convert the updated sum to a string and create a new
packet to send the sum back to the client
                String responseString = Integer.toString(sum);
                DatagramPacket responsePacket = new
DatagramPacket(responseString.getBytes(),
                        responseString.getBytes().length,
requestPacket.getAddress(), requestPacket.getPort());
                // Send the packet containing the updated sum to the client
                serverSocket.send(responsePacket);
                System.out.println("Returning sum of " + sum + " to client");
                System.out.println(" ");
            }
        } catch (SocketException e) {
            System.out.println("Socket Exception: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("IO Exception: " + e.getMessage());
        } finally {
            if (serverSocket != null) {
                serverSocket.close();
            }
        }
    }
}
```

🏁**Take a screenshot of your client console screen; Project2Task2ClientConsole**





🏁**Take a screenshot of your server console screen; Project2Task2ServerConsole**

Run: AddingServerUDP AddingClientUDP

```
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Con
Server started
Adding: 1 to 0
Returning sum of 1 to client

Adding: 2 to 1
Returning sum of 3 to client

Adding: -3 to 3
Returning sum of 0 to client

Adding: 4 to 0
Returning sum of 4 to client

Adding: 5 to 4
Returning sum of 9 to client

Adding: 6 to 9
```

Structure Bookmarks

Version Control Run TODO Problems Terminal Profiler

Run: AddingServerUDP AddingClientUDP

```
Adding: 5 to 4
Returning sum of 9 to client

Adding: 6 to 9
Returning sum of 15 to client

Adding: 7 to 15
Returning sum of 22 to client

Adding: -8 to 22
Returning sum of 14 to client

Adding: 9 to 14
Returning sum of 23 to client

Adding: 10 to 23
Returning sum of 33 to client
```

Structure Bookmarks

Version Control Run TODO Problems Terminal

All files are up-to-d... (2 minutes ago)

Project2Task3:

🏁Project2Task3Client

```
package edu.cmu.ds;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
import java.util.Scanner;

public class RemoteVariableClientUDP {
    /**
     * Client component that sends integer values to the server,
     * receives the sum of those integers computed by the server,
     * and displays the result to the user.
     * The client reads user input from the console,
     * and sends requests to the server by creating a DatagramPacket
containing the user input as a string,
     * and sends it to the server at the specified port. The server receives
the packet, parses the integer value,
     * computes the sum, and sends it back to the client. The client then
receives the response from the server
     * and displays it to the user.
     */
    public static void main(String[] args) {
        // Creates a new Scanner object to read input from the user
        Scanner scanner = new Scanner(System.in);

        int serverPort = 0;
        try {
            System.out.println("The client is running.");
            System.out.print("Please enter server port: \n");
            // Read the server port number from the user input
            serverPort = Integer.parseInt(scanner.nextLine());
        } catch (NumberFormatException e) {
            System.out.println("Invalid port number: " + e.getMessage());
            System.exit(1);
        }
        System.out.println(" ");
        // prompts the user to enter the server port number,
        // reads the input from the user, and stores the value in the
"serverPort" variable.
        // If the user enters an invalid value, the program prints an error
message and exits.
        // Loop until user enters "halt!"
        while (true) {
            try {
                // Read input from the user
                System.out.println("1. Add a value to your sum.");
```

```
                System.out.println("2. Subtract a value from your sum.");
                System.out.println("3. Get your sum.");
                System.out.println("4. Exit client.");
                System.out.print("Please enter your choice: \n");
                int choice = Integer.parseInt(scanner.nextLine());
                if (choice < 1 || choice > 4) {
                    System.out.println("Invalid input. Please enter a number
between 1 and 4.");
                    continue;
                }
                if (choice == 4) {
                    System.out.println("Client side quitting. The remote
variable server is still running.");
                    break;
                }
                int value = 0;
                if (choice != 3) {
                    System.out.print("Enter value: \n");
                    value = Integer.parseInt(scanner.nextLine());
                }
                System.out.print("Enter your ID: \n");
                int id = Integer.parseInt(scanner.nextLine());
                // create message
                String message = id + ",";

                if (choice == 1) {
                    message += "add," + value;
                } else if (choice == 2) {
                    message += "subtract," + value;
                } else if (choice == 3) {
                    message += "get";
                }
                sendMessage(message, serverPort);
            } catch (NumberFormatException e) {
                System.out.println("Invalid input: " + e.getMessage());
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        }
        scanner.close();
    }

    /**
     * Send message to server and receive the reply and print.
     * @param message
     * @param serverPort
     * @throws IOException
     */
    public static void sendMessage(String message, int serverPort) throws
IOException {
        //  Declare and initialize a DatagramSocket variable
        DatagramSocket clientSocket = null;
        try {
            // Create a new DatagramSocket
            clientSocket = new DatagramSocket();
            // Socket communication with the server
            final int BUFFER_SIZE = 1000;
```

```
            // Send message
            byte[] messageBytes = message.getBytes();
            // Get the server address
            InetAddress serverAddress = InetAddress.getByName("localhost");
            // Create a DatagramPacket to send the message
            DatagramPacket request = new DatagramPacket(messageBytes,
messageBytes.length, serverAddress, serverPort);
            clientSocket.send(request);
            // receive response
            byte[] buffer = new byte[1000];
            // Create a DatagramPacket to receive the response
            DatagramPacket reply = new DatagramPacket(buffer, buffer.length);
            clientSocket.receive(reply);
            // display response
            String response = new String(reply.getData(), 0,
reply.getLength());
            System.out.println("The result is " + response + ".\n");
        } catch (SocketException e) {
            System.out.println("Socket: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("IO: " + e.getMessage());
        } finally {
            if (clientSocket != null)
                clientSocket.close();
        }
    }
}
```

🏁Project2Task3Server

```
package edu.cmu.ds;

import java.io.*;
import java.net.*;
import java.util.Iterator;
import java.util.Map;
import java.util.Scanner;
import java.util.TreeMap;

public class RemoteVariableServerUDP {
    /**
     * This is the server class for the remote variable program using UDP
protocol.
     * The server listens for incoming packets from clients,
     * calculates and updates the sum of integer inputs for each client,
     * and sends back the updated sum as a response.
     */
    // The server listens on this port for incoming requests from clients
    final static int SERVER_PORT = 6789;
    // The maximum size of the byte array used for packet transmission
    final static int BUFFER_SIZE = 1000;
    // The current sum of all integer inputs received from clients
    static int sum = 0;
    // The buffer used to store incoming and outgoing packets
    static byte[] buffer = new byte[BUFFER_SIZE];
```

```java
    // The UDP socket used to listen for incoming packets
    static DatagramSocket serverSocket = null;

    /**
     * The main method that starts the server and listens for incoming
packets from clients.
     * @param args
     */
    public static void main(String[] args) {
        // A map to store the sum for each client
        Map<Integer, Integer> sums = new TreeMap<Integer, Integer>();
        try {
            // Create a new scanner to read user input
            Scanner scanner = new Scanner(System.in);
            // Create a new UDP socket to listen for incoming packets
            serverSocket = new DatagramSocket(SERVER_PORT);
            System.out.println("Server started");
            // Listen for incoming packets indefinitely
            while (true) {
                // Create a new packet to store incoming data
                DatagramPacket requestPacket = new DatagramPacket(buffer,
BUFFER_SIZE);
                // Wait for incoming packets
                serverSocket.receive(requestPacket);
                // Extract the value from the incoming packet data
                String message = new String(requestPacket.getData(), 0,
requestPacket.getLength());
                String[] parts = message.split(",");
                int id = Integer.parseInt(parts[0]);
                String operation = parts[1];
                int value = 0;
                // If the operation is add or subtract, extract the value
from the message
                if (operation.equals("add") || operation.equals("subtract"))
{
                    value = Integer.parseInt(parts[2]);
                }
                // Calculate the sum
                if (sums.containsKey(id)) {
                    sum = sums.get(id);
                }
                if (operation.equals("add")) {
                    sum += value;
                } else if (operation.equals("subtract")) {
                    sum -= value;
                }
                // Update the map with the new sum
                sums.put(id, sum);
                // Prepare the response
                byte[] responseBuffer;
                responseBuffer = Integer.toString(sum).getBytes();
                DatagramPacket response = new DatagramPacket(responseBuffer,
responseBuffer.length,
                        requestPacket.getAddress(), requestPacket.getPort());
                serverSocket.send(response);
                // Print the Tree Map
                System.out.println("Print the Tree Map");
```

```
                Iterator iter = sums.keySet().iterator();
                while (iter.hasNext()) {
                    Object key = iter.next();
                    Object val = sums.get(key);
                    System.out.println("[" + key + "," + val + "]");
                }
                System.out.println("\n");
                sum = 0;
            }
        } catch (SocketException e) {
            System.out.println("Socket Exception: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("IO Exception: " + e.getMessage());
        } finally {
            if (serverSocket != null) {
                serverSocket.close();
            }
        }
    }
}
```

🏁Project2Task3ClientConsole

```
Run:  RemoteVariableClientUDP   RemoteVariableServerUDP
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java -javaagent:/Applications/Int
The client is running.
Please enter server port:
6789

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
1
Enter value:
5
Enter your ID:
10
The result is 5.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
2
Enter value:
10
Enter your ID:
10
The result is -5.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
3
Enter your ID:
10
The result is -5.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
4
Client side quitting. The remote variable server is still running.

Process finished with exit code 0
```

Run: RemoteVariableClientUDP × RemoteVariableServerUDP ×

```
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java -javaagent:/Appl
The client is running.
Please enter server port:
6789

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
1
Enter value:
20
Enter your ID:
20
The result is 20.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
2
Enter value:
10
Enter your ID:
20
The result is 10.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
3
Enter your ID:
20
The result is 10.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
4
Client side quitting. The remote variable server is still running.

Process finished with exit code 0
```

Structure

Bookmarks

Run: RemoteVariableClientUDP RemoteVariableServerUDP

```
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java -javaagent:/A
The client is running.
Please enter server port:
6789

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
1
Enter value:
99
Enter your ID:
30
The result is 99.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
2
Enter value:
1024
Enter your ID:
30
The result is -925.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
3
Enter your ID:
30
The result is -925.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
4
Client side quitting. The remote variable server is still running.

Process finished with exit code 0
```

Structure

Bookmarks

Project2Task3ServerConsole

```
Run:  RemoteVariableClientUDP   RemoteVariableServerUDP
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java -j
Server started
Visitor's ID is: 10
The operation requested is: add
The value of the variable is: 5

Visitor's ID is: 10
The operation requested is: subtract
The value of the variable is: -5

Visitor's ID is: 10
The operation requested is: get
The value of the variable is: -5

Visitor's ID is: 20
The operation requested is: add
The value of the variable is: 20

Visitor's ID is: 20
The operation requested is: subtract
The value of the variable is: 10

Visitor's ID is: 20
The operation requested is: get
The value of the variable is: 10

Visitor's ID is: 30
The operation requested is: add
The value of the variable is: 99

Visitor's ID is: 30
The operation requested is: subtract
The value of the variable is: -925

Visitor's ID is: 30
The operation requested is: get
The value of the variable is: -925
```

Project2Task4:

Project2Task4Client

```
package edu.cmu.ds;

import java.io.*;
import java.net.*;
import java.util.Scanner;

public class EchoClientTCP {
    /**
     * Client component that sends integer values to the server,
     * receives the sum of those integers computed by the server,
     * and displays the result to the user.
     * The client reads user input from the console,
     * and sends requests to the server by creating a DatagramPacket
containing the user input as a string,
     * and sends it to the server at the specified port. The server receives
the packet, parses the integer value,
     * computes the sum, and sends it back to the client. The client then
receives the response from the server
     * and displays it to the user.
     */
    public static void main(String[] args) {
        Socket clientSocket = null;
        // Creates a new Scanner object to read input from the user
        Scanner scanner = new Scanner(System.in);

        int serverPort = 0;
        try {
            System.out.println("The client is running.");
            System.out.print("Please enter server port: \n");
            // Read the server port number from the user input
            serverPort = Integer.parseInt(scanner.nextLine());
            clientSocket = new Socket("localhost", serverPort);
            BufferedReader in = null;
            PrintWriter out = null;
            in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
            out = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(clientSocket.getOutputStream())));
            while (true) {
                try {
                    // Read input from the user
                    System.out.println("1. Add a value to your sum.");
                    System.out.println("2. Subtract a value from your sum.");
                    System.out.println("3. Get your sum.");
                    System.out.println("4. Exit client.");
                    System.out.print("Please enter your choice: \n");
                    int choice = Integer.parseInt(scanner.nextLine());
                    if (choice < 1 || choice > 4) {
```

```
                        System.out.println("Invalid input. Please enter a
number between 1 and 4.");
                        continue;
                    }
                    if (choice == 4) {
                        System.out.println("Client side quitting. The remote
variable server is still running.");
                        break;
                    }
                    int value = 0;
                    if (choice != 3) {
                        System.out.print("Enter value: \n");
                        value = Integer.parseInt(scanner.nextLine());
                    }
                    System.out.print("Enter your ID: \n");
                    int id = Integer.parseInt(scanner.nextLine());
                    // create message
                    String message = id + ",";

                    if (choice == 1) {
                        message += "add," + value;
                    } else if (choice == 2) {
                        message += "subtract," + value;
                    } else if (choice == 3) {
                        message += "get";
                    }
                    //send requestString to server
                    out.println(message);
                    out.flush();
                    // read a line of data from the stream
                    String data = in.readLine();
                    System.out.println("The result is " + data + "\n");
                } catch (NumberFormatException e) {
                    System.out.println("Invalid input: " + e.getMessage());
                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
            }
            scanner.close();
        } catch (NumberFormatException e) {
            System.out.println("Invalid port number: " + e.getMessage());
            System.exit(1);
        } catch (UnknownHostException e) {
            throw new RuntimeException(e);
        } catch (SocketException e) {
            System.out.println("Socket: " + e.getMessage());
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
        System.out.println(" ");
        // prompts the user to enter the server port number,
        // reads the input from the user, and stores the value in the
"serverPort" variable.
        // If the user enters an invalid value, the program prints an error
message and exits.
        // Loop until user enters "halt!"
    }
```

```
}
```

Project2Task4Server

```
package edu.cmu.ds;

import java.net.*;
import java.io.*;
import java.util.Map;
import java.util.Scanner;
import java.util.TreeMap;

/**
 *  This is the server class for the remote variable program using UDP
protocol.
 *  the server listens for incoming packets from clients,
 *  calculates and updates the sum of integer inputs for each client,
 *  and sends back the updated sum as a response.
 */

public class EchoServerTCP {
    public static void main(String args[]) {
        System.out.println("Server Running"); //indicate start of the server
        Socket clientSocket = null;
        Map<Integer, Integer> sums = new TreeMap<Integer, Integer>();
        try {
            int serverPort = 7777; // the server port we are using
            int sum = 0;
            // Create a new server socket
            ServerSocket listenSocket = new ServerSocket(serverPort);

            /*
             * Forever,
             *   wait for a new connection
             *   read a line from the socket
             */
            while (true) {
                /*
                 * Block waiting for a new connection request from a client.
                 * When the request is received, "accept" it, and the rest
                 * the tcp protocol handshake will then take place, making
                 * the socket ready for reading and writing.
                 */
                clientSocket = listenSocket.accept();
                // If we get here, then we are now connected to a client.

                // Set up "in" to read from the client socket
                Scanner in = new Scanner(clientSocket.getInputStream());
                // Set up "out" to write to the client socket
                while (in.hasNextLine()) {
                    String message = in.nextLine();
                    //retrieve id, operation, val from byte arr
```

```
                    String[] parts = message.split(",");
                    int id = Integer.parseInt(parts[0]);
                    String operation = parts[1];
                    int value = 0;
                    // If the operation is add or subtract, extract the value
from the message
                    if (operation.equals("add") ||
operation.equals("subtract")) {
                        value = Integer.parseInt(parts[2]);
                    }
                    // Calculate the sum
                    if (sums.containsKey(id)) {
                        sum = sums.get(id);
                    }
                    if (operation.equals("add")) {
                        sum += value;
                    } else if (operation.equals("subtract")) {
                        sum -= value;
                    }
                    // Update the map with the new sum
                    sums.put(id, sum);
                    // Print visitor's ID
                    System.out.println("Visitor's ID is: " + id);
                    // Print the operation requested
                    System.out.println("The operation requested is: " +
operation);
                    // Print the value of the variable
                    System.out.println("The value of the variable is: " + sum
+ "\n");
                    PrintWriter out = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(clientSocket.getOutputStream())));
                    out.println(sum);
                    out.flush();
                    sum = 0;
                }
            }
        } catch (IOException e) {
            System.out.println("IO Exception:" + e.getMessage());
        } finally {
            try {
                if (clientSocket != null) {
                    clientSocket.close();
                }
            } catch (IOException e) {
                // ignore exception on close
            }
        }
    }
}
```

Project2Task4ClientConsole

```
Run:  EchoServerTCP  RemoteVariableClientUDP
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java -javaa
The client is running.
Please enter server port:
7777
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
1
Enter value:
10
Enter your ID:
10
The result is 10

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
2
Enter value:
5
Enter your ID:
10
The result is 5

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
3
Enter your ID:
10
The result is 5

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
4
Client side quitting. The remote variable server is still running.


Process finished with exit code 0
```

Run: EchoServerTCP RemoteVariableClientUDP

```
The client is running.
Please enter server port:
7777
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
1
Enter value:
5
Enter your ID:
20
The result is 5

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
2
Enter value:
40
Enter your ID:
20
The result is -35

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
3
Enter your ID:
20
The result is -35

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
4
Client side quitting. The remote variable server is still running.


Process finished with exit code 0
```

Structure Bookmarks

Version Control Run Debug TODO Problems Terminal Profiler Services Build

Run: EchoServerTCP × RemoteVariableClientUDP ×

```
The client is running.
Please enter server port:
7777
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
1
Enter value:
2048
Enter your ID:
30
The result is 2048

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
2
Enter value:
7506
Enter your ID:
30
The result is -5458

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
3
Enter your ID:
30
The result is -5458

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
4
Client side quitting. The remote variable server is still running.


Process finished with exit code 0
```

Structure Bookmarks

Version Control Run Debug TODO Problems Terminal Profiler Services Build

Project2Task4ServerConsole

```
Run:  EchoServerTCP  RemoteVariableClientUDP
/Library/Java/JavaVirtualMachines/temurin-17.jdk
Server Running
Visitor's ID is: 10
The operation requested is: add
The value of the variable is: 10

Visitor's ID is: 10
The operation requested is: subtract
The value of the variable is: 5

Visitor's ID is: 10
The operation requested is: get
The value of the variable is: 5

Visitor's ID is: 20
The operation requested is: add
The value of the variable is: 5

Visitor's ID is: 20
The operation requested is: subtract
The value of the variable is: -35

Visitor's ID is: 20
The operation requested is: get
The value of the variable is: -35

Visitor's ID is: 30
The operation requested is: add
The value of the variable is: 2048

Visitor's ID is: 30
The operation requested is: subtract
The value of the variable is: -5458

Visitor's ID is: 30
The operation requested is: get
The value of the variable is: -5458
```

Project2Task5:

🏁 Project2Task5Client

```
import java.math.BigInteger;
import java.net.*;
import java.io.*;
import java.security.NoSuchAlgorithmException;
import java.util.Random;
import java.util.Scanner;
import java.security.MessageDigest;

/**
 * implements a client that connects to a server and sends requests
 * to perform arithmetic operations on a variable stored in the server.
 * The client encrypts its requests using a digital signature and sends them
to the server.
 * The server verifies the signature and performs the requested operation on
the variable.
 * The result is then sent back to the client for display.
 */
public class SigningClientTCP {
    // Initialize variables for input/output streams and key information
    static BufferedReader in;
    static PrintWriter out;
    static BigInteger n, e, d, id;

    public static void main(String args[]) {
        // Initialize variables for user input and key generation
        Socket clientSocket = null;
        Scanner scanner = new Scanner(System.in);

        // Inform user that the client is running and generate a key ID
        System.out.println("The client is running.");
        generateKeyID();

        // Request server port from user
        System.out.print("Please enter server port: \n");
        int serverPort = Integer.parseInt(scanner.nextLine());

        try {
            // Connect to the server using the specified port number
            clientSocket = new Socket("localhost", serverPort);
            String message;

            // Initialize input and output streams
            System.out.println();
            in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
            out = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(clientSocket.getOutputStream())));

            while (true) {
                // Read input from the user
                System.out.println("1. Add a value to your sum.");
```

```
                System.out.println("2. Subtract a value from your sum.");
                System.out.println("3. Get your sum.");
                System.out.println("4. Exit client.");
                System.out.print("Please enter your choice: \n");
                int choice = Integer.parseInt(scanner.nextLine());

                // Check if user input is valid and prompt for new input if
not
                if (choice < 1 || choice > 4) {
                    System.out.println("Invalid input. Please enter a number
between 1 and 4.");
                    continue;
                }

                // If user selects option 4, inform them that the client is
quitting and break out of loop
                if (choice == 4) {
                    System.out.println("Client side quitting. The remote
variable server is still running.");
                    break;
                }

                // Prompt user for value to add/subtract and build message
string
                int value = 0;
                if (choice != 3) {
                    System.out.print("Enter value: \n");
                    value = Integer.parseInt(scanner.nextLine());
                }
                message = id + "," + e + "," + n + ",";

                // Determine operation selected by user and add corresponding
information to message string
                if (choice == 1) {
                    message += "add," + value;
                } else if (choice == 2) {
                    message += "subtract," + value;
                } else if (choice == 3) {
                    message += "get,";
                }

                // Sign message and append signature to end of message string
                String encryptedMessage = message + "/" + sign(message);

                // Send message to server
                out.println(encryptedMessage);
                out.flush();

                // Receive response from server and print result
                String data = in.readLine();
                System.out.println("The result is " + data + "\n");
            }
        } catch (IOException | NoSuchAlgorithmException e) {
            // Print exception message if one occurs
            System.out.println("IO Exception:" + e.getMessage());
        } finally {
            try {
```

```
                // Close the client socket if it is not null
                if (clientSocket != null) {
                    clientSocket.close();
                }
            } catch (IOException e) {
                // Ignore exception on close
            }
        }
    }

    // Generate the public key and private key everytime the client
    public static void generateKeyID(){
        System.out.println("Generating keys");
        Random rnd = new Random();
        // Step 1: Generate two large random primes.
        // We use 400 bits here, but best practice for security is 2048 bits.
        // Change 400 to 2048, recompile, and run the program again, and you
will
        // notice it takes much longer to do the math with that many bits.
        BigInteger p = new BigInteger(400, 100, rnd);
        BigInteger q = new BigInteger(400, 100, rnd);

        // Step 2: Compute n by the equation n = p * q.
        n = p.multiply(q);

        // Step 3: Compute phi(n) = (p-1) * (q-1)
        BigInteger phi =
(p.subtract(BigInteger.ONE)).multiply(q.subtract(BigInteger.ONE));

        // Step 4: Select a small odd integer e that is relatively prime to
phi(n).
        // By convention the prime 65537 is used as the public exponent.
        e = new BigInteger("65537");

        // Step 5: Compute d as the multiplicative inverse of e modulo
phi(n).
        d = e.modInverse(phi);

        String publicKey, privateKey;

        String public_key = "(" + e + "," + n + ")";
        String private_key = "(" + d + "," + n + ")";
        System.out.println("your public key = " + public_key);  // Step 6:
(e,n) is the RSA public key
        System.out.println("your private key = " + private_key);  // Step 7:
(d,n) is the RSA private key
        // Generate ID
        try {
            MessageDigest md = MessageDigest.getInstance("SHA-256");
            md.update(public_key.getBytes());
            byte[] hash_value = md.digest();
            byte[] id_byte = new byte[20];
            // copy the last 20 bytes to id_byte
            for(int i = 0; i < 20; i++){
                id_byte[20 - i - 1] = hash_value[hash_value.length - i - 1];
            }
            id = new BigInteger(id_byte);
```

```
            System.out.println("Your id is: " + id);
        }
        catch(NoSuchAlgorithmException e) {
            System.out.println("No Hash available" + e);
        }
    }


    static public String sign(String message) throws
UnsupportedEncodingException, NoSuchAlgorithmException {
        // compute the digest with SHA-256
        byte[] bytesOfMessage = message.getBytes("UTF-8");
        MessageDigest md = MessageDigest.getInstance("SHA-256");
        byte[] bigDigest = md.digest(bytesOfMessage);
        // we only want two bytes of the hash for ShortMessageSign
        // we add a 0 byte as the most significant byte to keep
        // the value to be signed non-negative.
        byte[] messageDigest = new byte[bigDigest.length + 1];
        messageDigest[0] = 0;   // most significant set to 0
        for(int i = 0; i < bigDigest.length; i++){
            messageDigest[i+1] = bigDigest[i]; // take a byte from SHA-256
        }
        // From the digest, create a BigInteger
        BigInteger m = new BigInteger(messageDigest);
        // encrypt the digest with the private key
        BigInteger c = m.modPow(d, n);
        // return this as a big integer string
        return c.toString();
    }
}
```

🏁 Project2Task5Server

```
import java.math.BigInteger;
import java.net.*;
import java.io.*;
import java.security.MessageDigest;
import java.util.Map;
import java.util.Scanner;
import java.util.TreeMap;

/**
 * implements a simple server that performs addition and subtraction
operations on numbers.
 * It listens on a port, accepts TCP connections from clients,
 * and performs the requested operation on a shared variable.
 * The server uses public-key encryption to ensure that requests are
authorized.
 */
public class VerifyingServerTCP {
    static BigInteger n, e, id; // server's public key and client's ID
    static String operation; // operation requested by the client
    static Map<BigInteger, Integer> sums = new TreeMap<>(); // shared
variable storing visitor's sum
    static PrintWriter out; // output stream to the client
```

```
    static Scanner in; // input stream from the client

    public static void main(String args[]) {
        Socket clientSocket = null;
        try {
            int serverPort = 7777;
            System.out.println("Server Running"); // indicate start of the
server
            ServerSocket listenSocket = new ServerSocket(serverPort);
            clientSocket = listenSocket.accept(); // accept the first client
connection
            in = new Scanner(clientSocket.getInputStream());
            out = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(clientSocket.getOutputStream())));

            while (true) {
                String data;
                if(in.hasNextLine()){
                    // read the client's message
                    data = in.nextLine();
                    // parse the message, which is in the format
"id,e,n,operation,number/signature"
                    String[] message = data.split("/");
                    String[] parts = message[0].split(",");
                    id = new BigInteger(parts[0]);
                    e = new BigInteger(parts[1]);
                    n = new BigInteger(parts[2]);
                    operation = parts[3];
                    int number = 0;
                    try{
                        if (parts[4].length() != 0) {
                            number = Integer.parseInt(parts[4]);
                        }
                    } catch (ArrayIndexOutOfBoundsException e) {
                    }
                    // verify the signature of the message using the client's
public key
                    if(!idMatch(message[0], message[1])){
                        // if the signature is invalid, terminate the
connection
                        out.println("Request encryption error");
                        out.flush();
                        break;
                    }
                    // perform the requested operation on the shared variable
                    add(operation, id, number);
                }
                else {
                    // if there is no more data from the client, accept the
next client connection
                    clientSocket = listenSocket.accept();
                    in = new Scanner(clientSocket.getInputStream());
                    out = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(clientSocket.getOutputStream())));
                }
            }
        } catch (IOException e) {
```

```
            System.out.println("IO Exception:" + e.getMessage());
        } finally {
            // close the connection when the client terminates
            try {
                if (clientSocket != null) {
                    clientSocket.close();
                }
            } catch (IOException e) {
                // ignore exception on close
            }
        }
    }

    /**
     * Adds, subtracts, or gets the sum of the visitor identified by the
given ID.
     * @param operation the operation to perform (add, subtract, or get)
     * @param id the ID of the visitor
     * @param number the number to add or subtract (ignored for get)
     * @return the current sum of the visitor
     */
    public static int add(String operation, BigInteger id, int number){
        System.out.println("Visitor's ID is: " + id);
        System.out.println("The operation requested is:" + operation);
        // get the current sum of the visitor
        int value = sums.getOrDefault(id,0);
        // update the sum based on the requested operation
        if(operation.equals("add")){
            sums.put(id, value + number);
        }
        else if (operation.equals("subtract")){
            sums.put(id, value - number);
        } else if (operation.equals("get")) {
            sums.put(id, value);
        }
        value = sums.getOrDefault(id,0);
        System.out.println("The value of the variable is: " + value +"\n");
        // return the number after calculation
        out.println(value);
        out.flush();
        return sums.getOrDefault(id,0);
    }

    public static boolean idMatch(String messageToCheck, String
encryptedHashStr) {
        BigInteger decryptedHash = null;
        BigInteger bigIntegerToCheck = null;
        try{
            // Decrypt it
            decryptedHash = new BigInteger(encryptedHashStr).modPow(e, n);
            // Get the bytes from messageToCheck
            byte[] bytesOfMessageToCheck = messageToCheck.getBytes("UTF-8");
            // compute the digest of the message with SHA-256
            MessageDigest md = MessageDigest.getInstance("SHA-256");

            byte[] bigDigest = md.digest(bytesOfMessageToCheck);
```

```
            // messageToCheckDigest is a full SHA-256 digest
            // add a zero byte in front of bigDigest
            byte[] messageToCheckDigest  = new byte[bigDigest.length + 1];
            messageToCheckDigest [0] = 0;   // most significant set to 0
            for(int i = 0; i < bigDigest.length; i++){
                messageToCheckDigest [i+1] = bigDigest[i];
            }
            bigIntegerToCheck = new BigInteger(messageToCheckDigest);
        } catch (Exception e){
            e.printStackTrace();
        }
        System.out.println("Client hash message: " + decryptedHash);
        System.out.println("Hash value of the message:" + bigIntegerToCheck);
        System.out.println("Verify result: " +
bigIntegerToCheck.equals(decryptedHash));
        return bigIntegerToCheck.equals(decryptedHash);
    }
}
```

🏁 Project2Task5ClientConsole

Run: SigningClientTCP VerifyingServerTCP

```
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java -javaagent:/Ap
The client is running.
Generating keys
your public key = (65537,29169247275889627971963988837984488976979506695352152373834 58
your private key = (22125846782413372381132395640842347315617212213548777008530522315 6
Your id is: -350876654367165105893655060812126990667834842089
Please enter server port:
7777

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
1
Enter value:
5
The result is 5

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
2
Enter value:
10
The result is -5

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
3
The result is -5

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
4
Client side quitting. The remote variable server is still running.

Process finished with exit code 0
```

Structure Bookmarks

Version Control Run TODO Problems Terminal Profiler Services Build

Run: SigningClientTCP VerifyingServerTCP

```
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java -javaagent:/Applications
The client is running.
Generating keys
your public key = (65537,262448646654083226850331430578706062082675162046846070824480929486507 92
your private key = (159670910163596234508390905901005052220278988374991936401020852655538766986 9
Your id is: -699938994834898388856442059453463465345953462528
Please enter server port:
7777

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
1
Enter value:
20
The result is 20

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
2
Enter value:
99
The result is -79

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
3
The result is -79

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
4
Client side quitting. The remote variable server is still running.

Process finished with exit code 0
```

Version Control | Run | TODO | Problems | Terminal | Profiler | Services | Build

Structure | Bookmarks

Run: SigningClientTCP VerifyingServerTCP

```
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java -javaagent:/Applica
The client is running.
Generating keys
your public key = (65537,267290338177900933536342051416502884203285500341935357183397822959
your private key = (191059755287577202686169335499908358841373766712213005517699229661842 94
Your id is: 550616311977784329949997611655896442519395486 79
Please enter server port:
7777

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
1
Enter value:
1024
The result is 1024

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
2
Enter value:
99
The result is 925

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
3
The result is 925

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client.
Please enter your choice:
4
Client side quitting. The remote variable server is still running.

Process finished with exit code 0
```

Structure

Bookmarks

Project2Task5ServerConsole

```
Run:  SigningClientTCP  VerifyingServerTCP
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/C
Server Running
Client hash message: 110056723981977980009117655594331301988356407599098204057765270121978955403433
Hash value of the message:110056723981977980009117655594331301988356407599098204057765270121978955403433
Verify result: true
Visitor's ID is: -350876654367165105893655060812126990667834842089
The operation requested is:add
The value of the variable is: 5

Client hash message: 114072814034991720677716931245020096278707271540355527130827411917298021724 3
Hash value of the message:1140728140349917206777169312450200962787072715403555271308274119172980217243
Verify result: true
Visitor's ID is: -350876654367165105893655060812126990667834842089
The operation requested is:subtract
The value of the variable is: -5

Client hash message: 471888494868677385165768338931309850893440726948726014975390615459893057234 60
Hash value of the message:4718884948686773851657683389313098508934407269487260149753906154598930572346 0
Verify result: true
Visitor's ID is: -350876654367165105893655060812126990667834842089
The operation requested is:get
The value of the variable is: -5

Client hash message: 7284428019227361534403603344355072613620786256152435925124414827044452463 4109
Hash value of the message:72844280192273615344036033443550726136207862561524359251244148270444524634109
Verify result: true
Visitor's ID is: -699938994834898388856442059453463465345953462528
The operation requested is:add
The value of the variable is: 20

Client hash message: 5063948526010174022219712331312368340519090063962288991391572765324484617 5189
Hash value of the message:50639485260101740222197123313123683405190900639622889913915727653244846175189
Verify result: true
Visitor's ID is: -699938994834898388856442059453463465345953462528
The operation requested is:subtract
The value of the variable is: -79

Client hash message: 2695209161010742766125059937274753062468226044725852358244382832392277480883 8
Hash value of the message:26952091610107427661250599372747530624682260447258523582443828323922774808838
Verify result: true
Visitor's ID is: -699938994834898388856442059453463465345953462528
The operation requested is:get
The value of the variable is: -79

Client hash message: 6512779577485507730639359589631450309141978365383532991247325043563795825325 4
Hash value of the message:65127795774855077306393595896314503091419783653835329912473250435637958253254
Verify result: true
Visitor's ID is: 55061631197778432994999761165589644251939548679
The operation requested is:add
The value of the variable is: 1024

Client hash message: 11896853214286096360530123898113083246831870358617255949302636059771098438070
Hash value of the message:11896853214286096360530123898113083246831870358617255949302636059771098438070
```

Version Control | Run | TODO | Problems | Terminal | Profiler | Services | Build

Run: SigningClientTCP VerifyingServerTCP

```
The operation requested is:subtract
The value of the variable is: -5

Client hash message: 471888494868677385165768338931309850893440726948726014975390615459893057234 60
Hash value of the message:4718884948686773851657683389313098508934407269487260149753906154598930572346 0
Verify result: true
Visitor's ID is: -3508766543671651058936550608121269906678348420 89
The operation requested is:get
The value of the variable is: -5

Client hash message: 728442801922736153440360334435507261362078625615243592512441482704445246341 09
Hash value of the message:7284428019227361534403603344355072613620786256152435925124414827044452463410 9
Verify result: true
Visitor's ID is: -699938994834898388856442059453463465345953462528
The operation requested is:add
The value of the variable is: 20

Client hash message: 506394852601017402221971233131236834051909006396228899139157276532448461751 89
Hash value of the message:5063948526010174022219712331312368340519090063962288991391572765324484617518 9
Verify result: true
Visitor's ID is: -699938994834898388856442059453463465345953462528
The operation requested is:subtract
The value of the variable is: -79

Client hash message: 269520916101074276612505993727475306246822604472585235824438283239227748088 38
Hash value of the message:2695209161010742766125059937274753062468226044725852358244382832392277480883 8
Verify result: true
Visitor's ID is: -699938994834898388856442059453463465345953462528
The operation requested is:get
The value of the variable is: -79

Client hash message: 651277957748550773063935958963145030914197836538353299124732504356379582532 54
Hash value of the message:6512779577485507730639359589631450309141978365383532991247325043563795825325 4
Verify result: true
Visitor's ID is: 550616311977784329949997611655896442519395486 79
The operation requested is:add
The value of the variable is: 1024

Client hash message: 118968532142860963605301238981130832468318703586172559493026360597710984380 70
Hash value of the message:1189685321428609636053012389811308324683187035861725594930263605977109843807 0
Verify result: true
Visitor's ID is: 550616311977784329949997611655896442519395486 79
The operation requested is:subtract
The value of the variable is: 925

Client hash message: 524784857817619334983525209455281922740190301743429317362546005233422250576 90
Hash value of the message:5247848578176193349835252094552819227401903017434293173625460052334222505769 0
Verify result: true
Visitor's ID is: 550616311977784329949997611655896442519395486 79
The operation requested is:get
The value of the variable is: 925
```

Version Control | Run | TODO | Problems | Terminal | Profiler | Services | Build